$$X^*(0, k_2^*) = \sum_{k_2} \left[\sum_{k_1} X(k_1, k_2) \right] \omega^{k_2^* k_2}.$$

Step (1) is a set of p 'polynomial transforms' involving no multiplications; step (2) consists of p DFTs on p points each since if

$$T_{k_2^*}(z) = \sum_{k_1} Y_{k_2^*}(k_1) z^{k_1}$$

then

$$T_{k_2^*}(\omega^{k_1^*}) = \sum_{k_1} Y_{k_2^*}(k_1) \omega^{k_1^* k_1} = Y_{k_2^*}^*(k_1^*);$$

step (3) is a permutation; and step (4) is a p-point DFT. Thus the 2D DFT on $p \times p$ points, which takes $2p$ p-point DFTs by the row–column method, involves only $(p + 1)$ p-point DFTs; the other DFTs have been replaced by polynomial transforms involving only additions.

This procedure can be extended to n dimensions, and reduces the number of 1D p-point DFTs from np^{n-1} for the row–column method to $(p^n - 1)/(p - 1)$, at the cost of introducing extra additions in the polynomial transforms.

A similar algorithm has been formulated by Auslander *et al.* (1983) in terms of Galois theory.

1.3.3.3.3. Global algorithm design

1.3.3.3.3.1. From local pieces to global algorithms

The mathematical analysis of the structure of DFT computations has brought to light a broad variety of possibilities for reducing or reshaping their arithmetic complexity. All of them are 'analytic' in that they break down large transforms into a succession of smaller ones.

These results may now be considered from the converse 'synthetic' viewpoint as providing a list of procedures for assembling them:

(i) the building blocks are one-dimensional p-point algorithms for p a small prime;

(ii) the low-level connectors are the multiplicative reindexing methods of Rader and Winograd, or the polynomial transform reindexing method of Nussbaumer and Quandalle, which allow the construction of efficient algorithms for larger primes p, for prime powers p^ν, and for p-primary pieces of shape $p^\nu \times \ldots \times p^\nu$;

(iii) the high-level connectors are the additive reindexing scheme of Cooley–Tukey, the Chinese remainder theorem reindexing, and the tensor product construction;

(iv) nesting may be viewed as the 'glue' which seals all elements.

The simplest DFT may then be carried out into a global algorithm in many different ways. The diagrams in Fig. 1.3.3.1 illustrate a few of the options available to compute a 400-point DFT. They may differ greatly in their arithmetic operation counts.

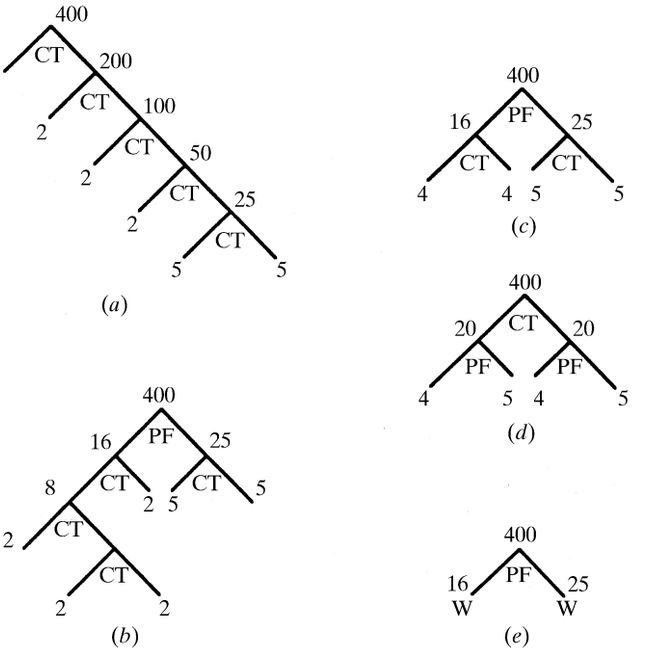

Fig. 1.3.3.1. A few global algorithms for computing a 400-point DFT. CT: Cooley–Tukey factorization. PF: prime factor (or Good) factorization. W: Winograd algorithm.

1.3.3.3.3.2. Computer architecture considerations

To obtain a truly useful measure of the computational complexity of a DFT algorithm, its arithmetic operation count must be tempered by computer architecture considerations. Three main types of trade-offs must be borne in mind:

(i) reductions in floating-point (f.p.) arithmetic count are obtained by reindexing, hence at the cost of an increase in integer arithmetic on addresses, although some shortcuts may be found (Uhrich, 1969; Burrus & Eschenbacher, 1981);

(ii) reduction in the f.p. multiplication count usually leads to a large increase in the f.p. addition count (Morris, 1978);

(iii) nesting can increase execution speed, but causes a loss of modularity and hence complicates program development (Silverman, 1977; Kolba & Parks, 1977).

Many of the mathematical developments above took place in the context of single-processor serial computers, where f.p. addition is substantially cheaper than f.p. multiplication but where integer address arithmetic has to compete with f.p. arithmetic for processor cycles. As a result, the alternatives to the Cooley–Tukey algorithm hardly ever led to particularly favourable trade-offs, thus creating the impression that there was little to gain by switching to more exotic algorithms.

The advent of new machine architectures with vector and/or parallel processing features has greatly altered this picture (Pease, 1968; Korn & Lambiotte, 1979; Fornberg, 1981; Swartzrauber, 1984):

(i) *pipelining* equalizes the cost of f.p. addition and f.p. multiplication, and the ideal 'blend' of the two types of operations depends solely on the number of adder and multiplier units available in each machine;

(ii) integer address arithmetic is delegated to specialized arithmetic and logical units (ALUs) operating concurrently with the f.p. units, so that complex reindexing schemes may be used without loss of overall efficiency.

Another major consideration is that of data flow [see *e.g.* Nawab & McClellan (1979)]. Serial machines only have few registers and few paths connecting them, and allow little or no overlap between computation and data movement. New architectures, on the other hand, comprise banks of vector registers (or 'cache memory') besides the usual internal registers, and dedicated ALUs can service data transfers between several of them simultaneously and concurrently with computation.

In this new context, the devices described in Sections 1.3.3.2 and 1.3.3.3 for altering the balance between the various types of arithmetic operations, and reshaping the data flow during the computation, are invaluable. The field of machine-dependent DFT algorithm design is thriving on them [see *e.g.* Temperton (1983*a*,*b*,*c*, 1985); Agarwal & Cooley (1986, 1987)].